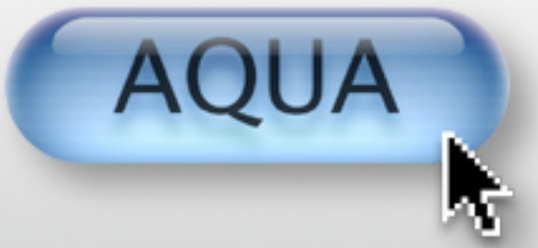
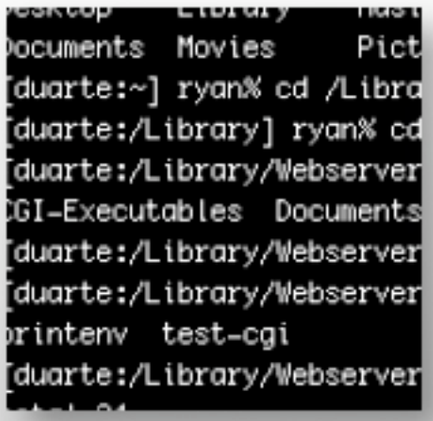
Audio Overview

Mac OS X Audio Rocks!

Session 407





Audio Overview

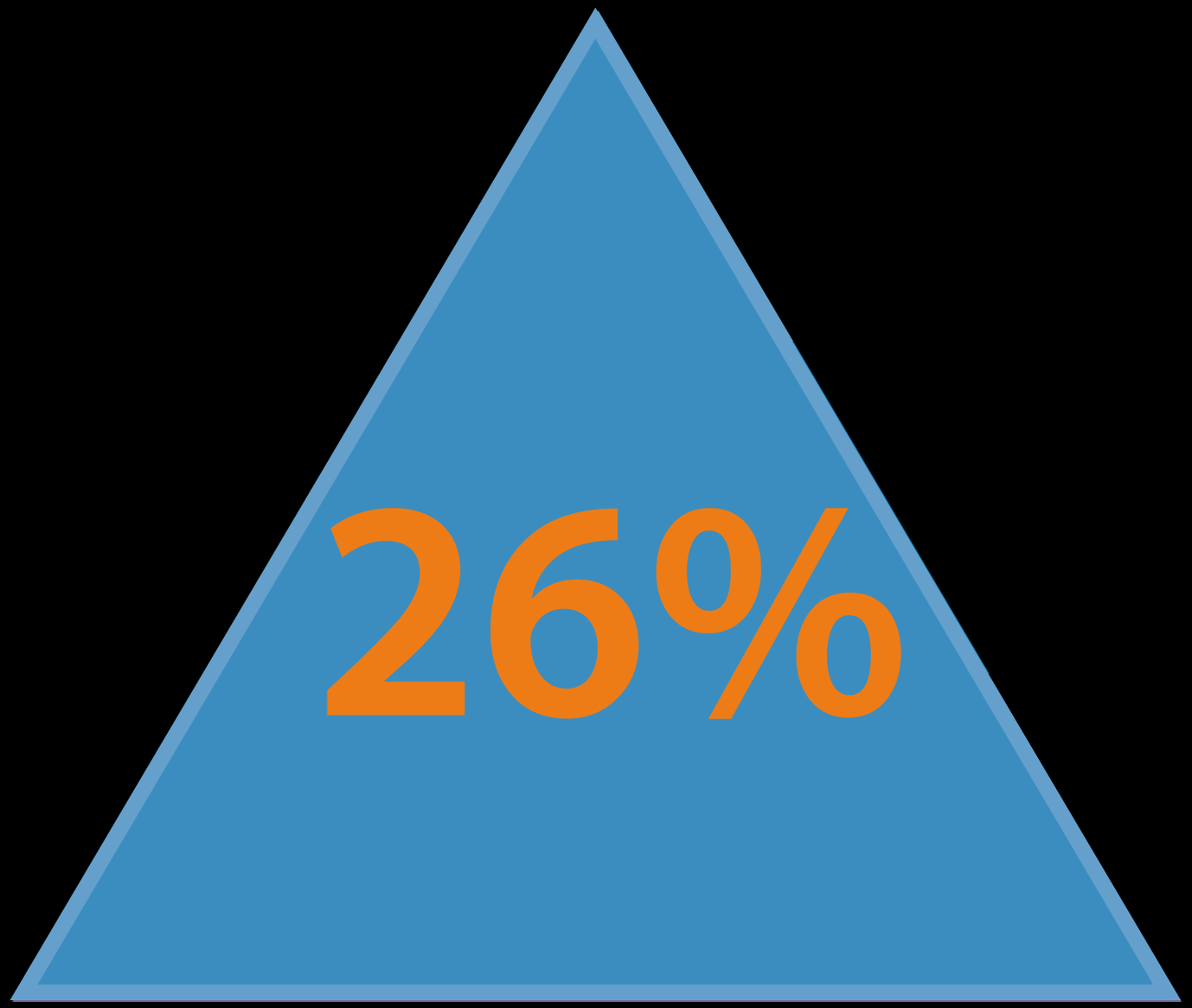
Mac OS X Audio Rocks!

Craig Linssen
Music and Audio Partnership Manager
craig@apple.com

What You'll Learn

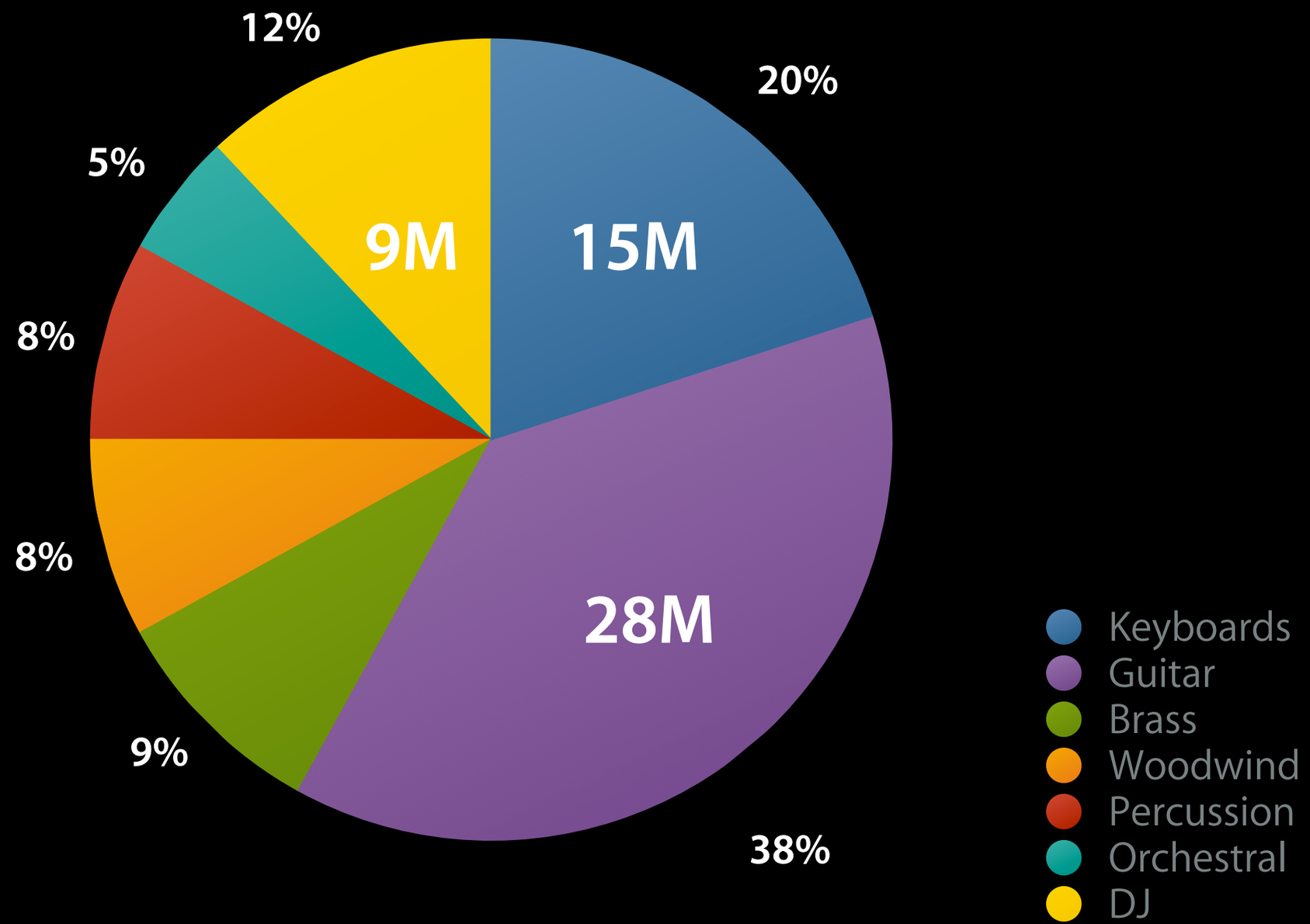
- Developer market opportunities
- Audio device driver changes in Panther
- Core audio architecture design objectives
- API's—How to get started!

Estimated Number of Music and Audio Creators in the U.S. Today



*Aggregate data from Gallup, American Music Conference/NAMM,Trade Magazine Surveys and 2000 US Census.

Breakdown of Music and Audio Markets in the U.S.



Mac OS X Core Audio Design Objectives

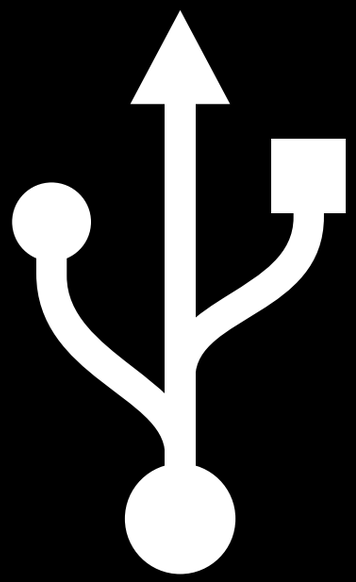
- Ease of use
  - Simplified user configuration
  - Streamlined development
- Performance
  - Professional requirements

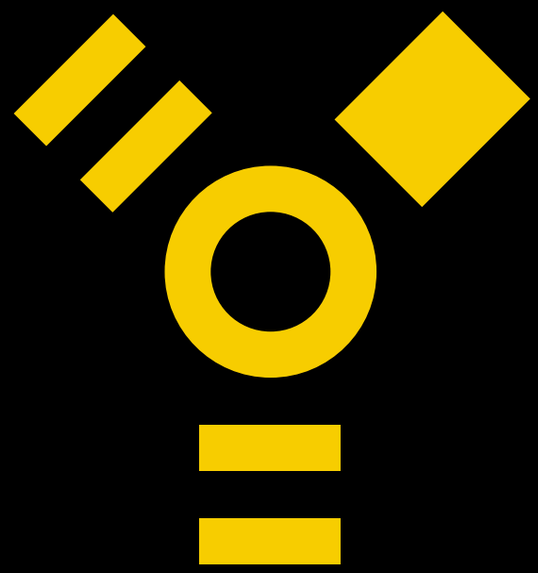
Mac OS X Audio Features

- The most feature rich audio platform!
- Core Audio is multi-channel!
  - High resolution
    - 24bit, 96khz and beyond!
- Low latency
- Native MIDI support
  - Audio MIDI setup

Class Drivers

- Plug-and-play device support





Audio Units and Audio Codecs

- Platform standards
- Extensibility!
- DSP plug-ins
- Virtual Instruments
- AAC, MP3, etc

Audio Unit Developers

Airy
AKAI
Antares
Apple
Arboretum
Arizona
Arturia
BIAS
Bitheadz
Bitshift
Celemony
Chris Reed
Destroy FX
Elemental Audio
Emagic
Expert Sleepers
FXpansion
Gmedia
Granted SW
Green Oak
Koen Tanghe
Illposed Software
Magnus Jonsson
Malcolm Haylock
Malte Steiner
Manyetas
mda
MOTU
Nao Tokui
Native Instruments
Novation
Ohm Force
reFX
Rogue Amoeba
Sagan Technology
The Sound Guy
Spectrasonics
Tassman
TC Works
Ultimate Sound Bank
Urs Heckmann
VirSyn
Waves
Wave Arts
Yellow Tools

Audio Drivers on Mac OS X

Nick Thompson
Engineering Manager, Audio CPU SW
nthompson@apple.com

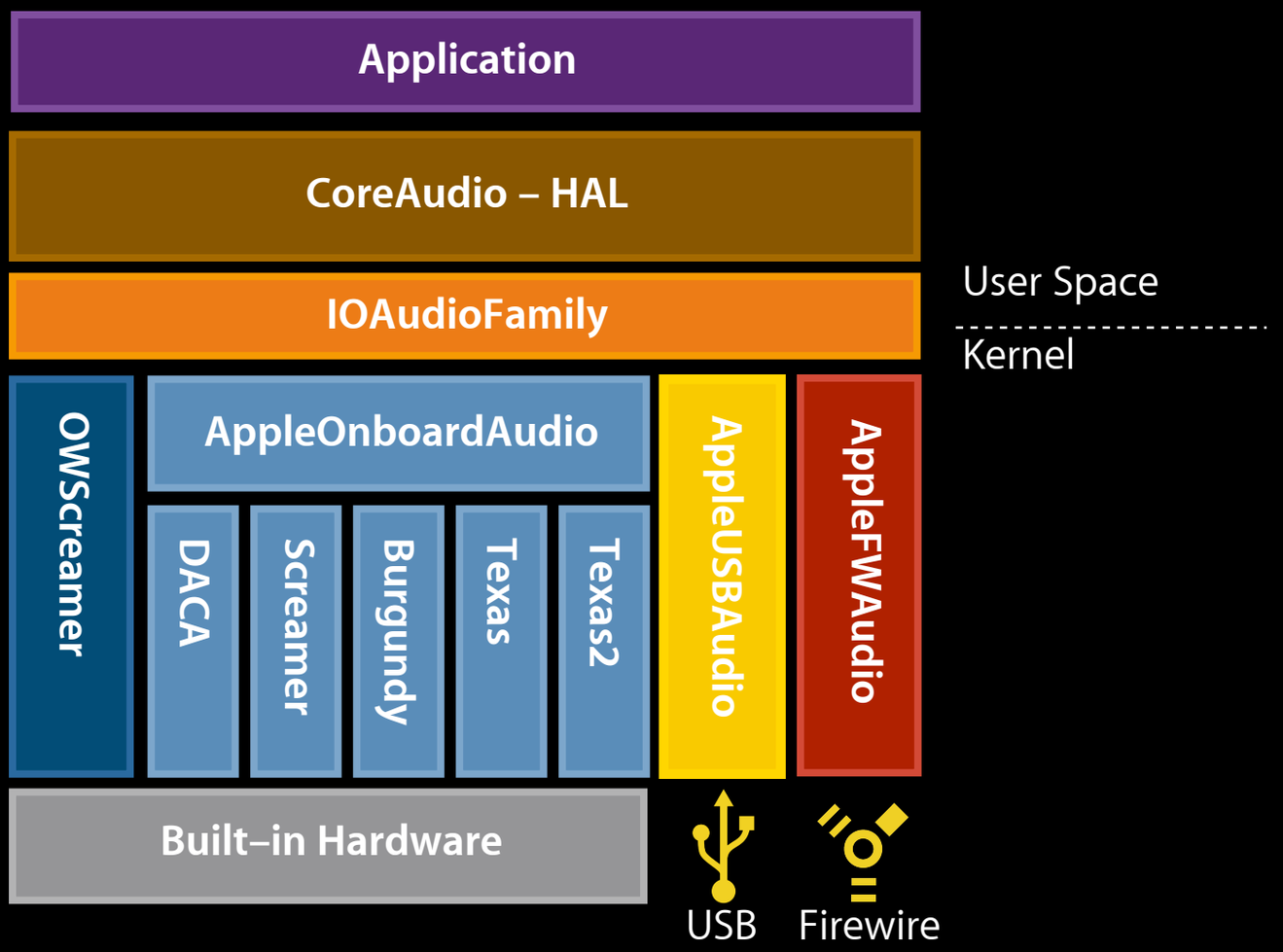
Mac OS X Kernel Based Audio Drivers

Interfacing the CoreAudio HAL to hardware

- Apple drivers in Darwin
- Audio on Power Macintosh G5 computers
  - Changes for PCI cards for the G5
  - Changes for IOAudioFamily drivers for the G5
- Apple Class Drivers
- Developer resources for writing device drivers

Audio Drivers on Mac OS X

Apple drivers in Darwin



- The interface for kernel-based audio drivers is IOAudioFamily
- Driver plug-ins reduce development time for built in hardware
- Apple develop class drivers for USB and FireWire 61883-6

Power Macintosh G5 Audio Features

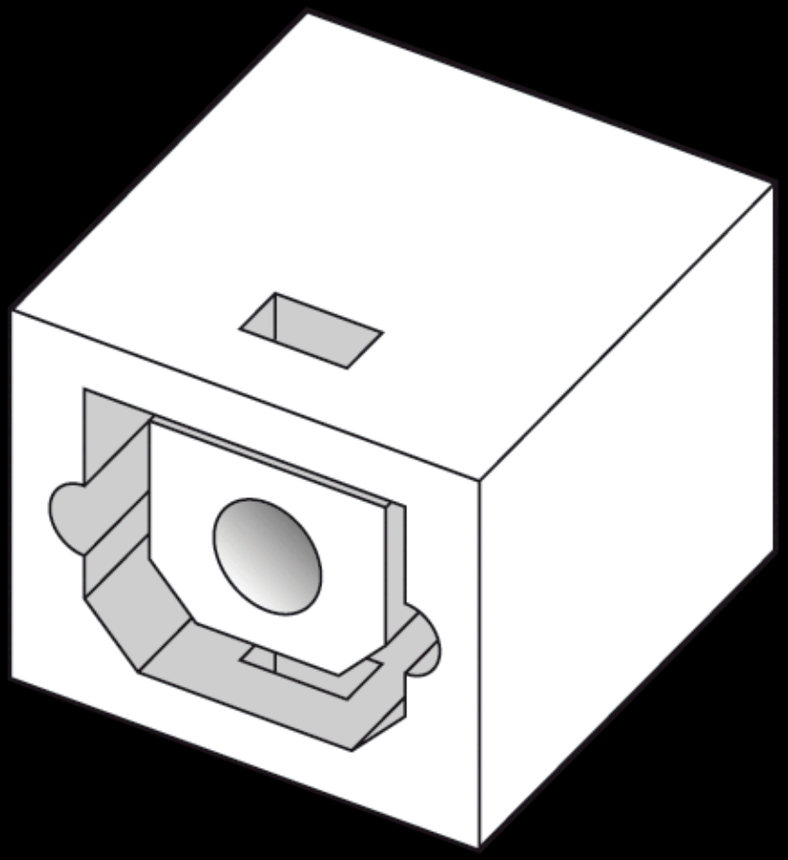
Radical departure for built in audio

- The Power Macintosh G5 computer is the first computer from Apple with built in digital I/O for audio
- The computer features analog I/O similar to previous tower implementations
- Some changes are needed for audio devices and device drivers

Power Macintosh G5 Audio Features

Input clock recovery and supported formats

- For digital input the clocking for the audio subsystem can be recovered from the incoming stream
  - This enables bit accurate copies of original source material
- Multi–format support enabled for new sample rates and sample sizes on both analog and digital parts
  - Sampling frequencies: 32kHz, 44.1kHz, and 48kHz
  - Sample sizes: 16 bit or 24 bit



– Cable uses JIS F-05 (friction lock) connectors as described by IEC 60874-17 (TOSLink)
– Data is transmitted in IEC 60958-3 (S/PDIF) format

G5 Audio Features

CPU implements a dual codec audio subsystem

- Analog section similar to previous tower systems
  - Multiple formats now supported
- Digital section supports both input and output

G5 Digital Audio

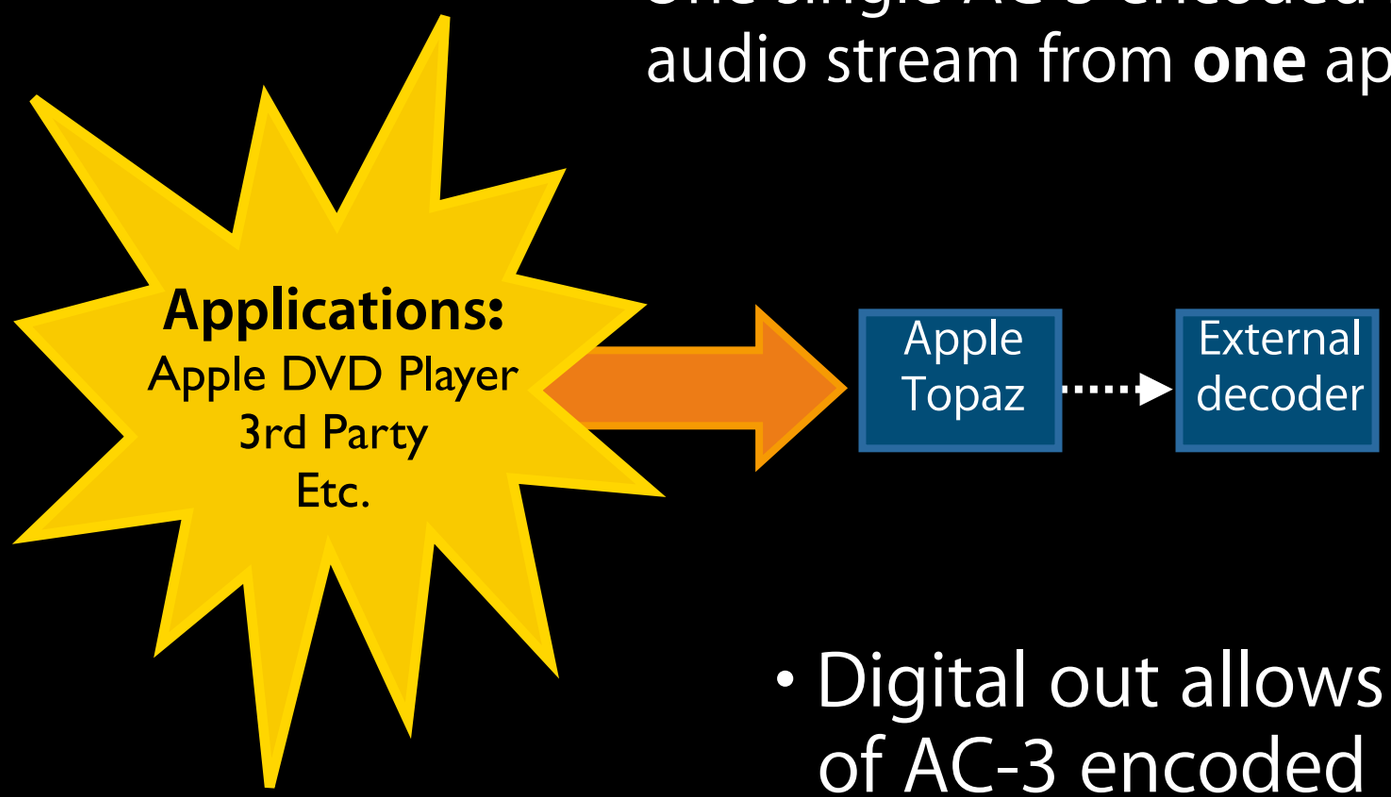
Digital Output and AC–3

PCM data versus encoded streams

- PCM stereo audio mixed from **all** apps

  or

- One single AC-3 encoded multi-channel audio stream from **one** application

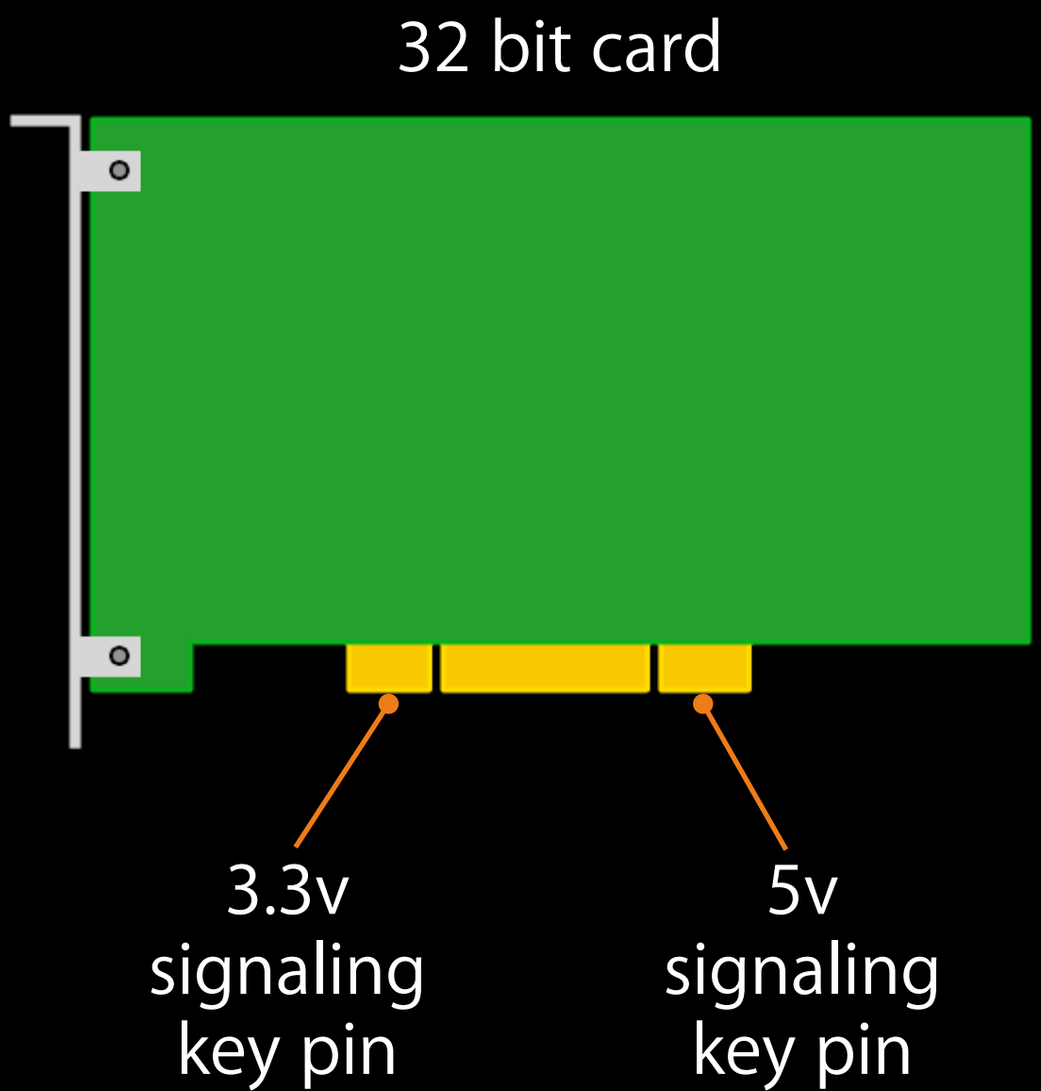


- Digital out allows streaming of AC-3 encoded multichannel data

“Hog Mode”

G5 and PCI cards

PCI card changes



- Physical changes
  - PCI cards that use 5v signaling are no longer supported
  - PCI cards need to be modified for, and keyed for, 3.3v signaling
  - Please take the opportunity to test your 3.3v and universal cards on the G5

Drivers and 64-bit Systems

IOAudioFamily driver changes

- Kernel drivers need to be made *"64 bit safe"*
- Avoid the following macros in `<mach/vm_param.h>`
  - `trunc_page()`, `round_page()`, `ptoa()`, `atop()`
  - These are ambiguous and silently incorrect when used with 64 bit addresses

- Instead use
  - `trunc_page_{32|64}`, `round_page_{32|64}`, `ptoa_{32|64}`, `atop_{32|64}`
  - Uses that operate on virtual addresses should be 32-bit
  - Uses that operate on physical addresses should be 64-bit

Drivers and 64 bit Systems

IOAudioFamily driver changes

- Remove assumptions about logical and physical addresses being the same
  - For example: AppleDBDMAAudio.kext was using `phystokv()` to map physical addresses to kernel virtual
  - This will not work in Apple products that support more than 2GB of RAM
    - These have a physical address space of more than 32 bits
  - Physical address space seen by the processor is not the same as the address space seen by the PCI card

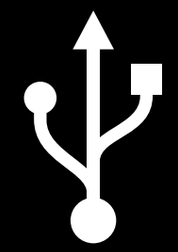
Drivers and 64-bit Systems

IOAudioFamily driver changes

- Make sure memory is prepared for I/O
  - Failure to do this may cause the driver to crash, since the contents of memory will not be found where expected
    - Preparing memory will cause a DART entry to be created ensuring mapping is correct

USB Class Driver

Spec-compliant devices



- Using Apple's class driver helps save development costs

  - Features
    - Adaptive sample rate (the driver will adapt to the sample rate for the device)
    - Any number of USB devices can be used simultaneously (subject to bandwidth limitations)
    - Open sourced in Darwin, reasonably easy to adapt to vendor specific devices
  - 95% of what most USB audio devices require
  - Mixer and selector units are not supported

USB Audio Device 2.0 Specification

- Apple participating in the USB Audio Device Working Group (DWG) on the USB Audio Device 2.0 specification
  - Does not require USB 2.0 for transport
  - Support for both full speed (12Mbps) and high speed devices (480Mbps)
- We are interested in examining new USB Audio 2.0 devices from an engineering perspective
  - We can help with defining spec compliant device descriptors

FireWire Audio Driver

61883-6 compliant driver shipped in Jaguar

- Standards-based driver saves development costs and provides support for:
  - AVC devices
    - BridgeCo
    - Oxford Semiconductor
  - device support
    - ping from Yamaha, Korg, Otari, PreSonus, Kurzweil, Apogee, etc.
  - Wide range of silicon supported

FireWire Audio Driver

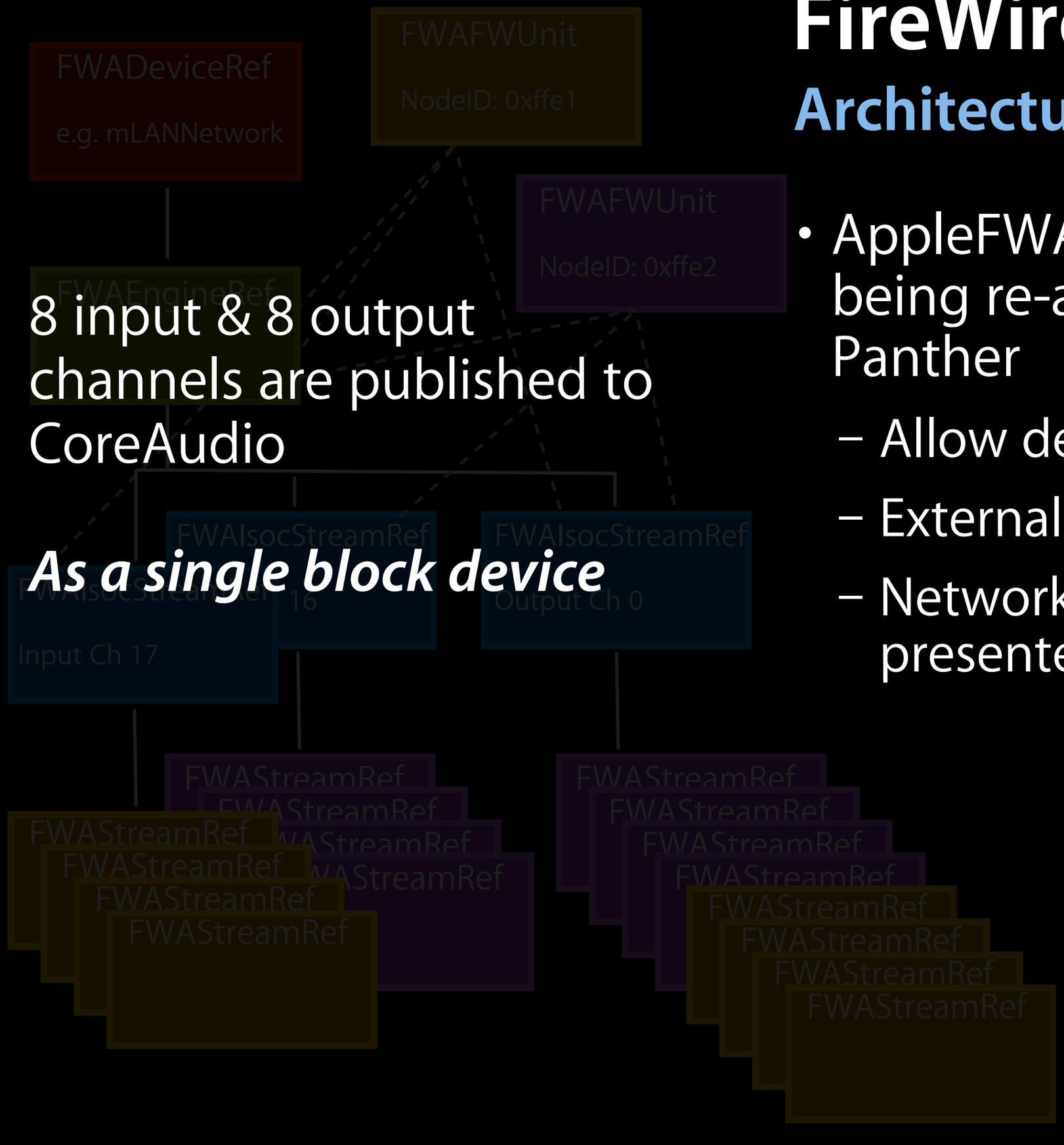
New for Panther

- Music sub-unit support
  - Input support
  - MIDI support
- Lower latency and Jitter
  - Target round trip latency <7mS
- External clocking
- Presentation of entire mLAN network as single entity in CoreAudio

FireWire Audio

Architecture for Panther

- AppleFWAudio kext is being re-architected for Panther
  - Allow device stacking
  - External sync
  - Networked devices presented as a single block



Developer Resources

Sample code for audio driver development

- Driver code in Darwin
  - IOAudioFamily
  - AppleUSBAudio
  - AppleOnboardAudio
- Sample code
  - Phantom audio driver
  - Sample PCI driver
- http://developer.apple.com/audio

Core Audio Overview

William Stewart
Core Audio Engineering
bills@apple.com

Introduction

- What is CoreAudio?
- Set of APIs for both Audio and MIDI services
  - Devices
  - Software processing
    - Plugins
    - Codecs/format conversions
  - Sequencing, general audio services

What You'll Learn

- What services does CoreAudio provide to applications to "Do Audio"

Connecting It All Together

MusicPlayer

AudioFile

MusicSequence

MusicTracks

Tempo

Events

AudioConverter

AudioCodec

AUGraph

DLSSynth

AUFilter

AU3DMixer

AUOutput

MIDIDevice

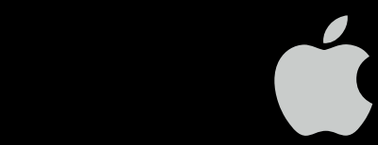
AudioDevice

Technology Framework

- API Set published in 4 Frameworks
  - CoreAudio
  - CoreMIDI
  - AudioUnit
  - AudioToolbox
- Dynamic Loading
  - AudioUnits
  - AudioCodecs
  - MIDI Drivers

CoreAudio Features

- Pro Audio requirements
  - Low latency and jitter
    - Audio and MIDI
  - High (no reasonable limit) sample rate
  - Multichannel aware throughout
    - Not just I/O
  - Extended resolution and rich data format abstractions



CoreAudio: HAL

- API to audio devices
  - Publishes device formats
  - Device status and management services
  - Device configuration
  - System device preferences

CoreMIDI

- API for transportation of MIDI data through the system
  - Input and output
    - Drivers
  - Inter-application
    - “Virtual” sources and destinations
    - IAC busses
- Configuration/publication of device characteristics

Demo

Audio MIDI Setup

AudioUnit

- Generalized plugin format for processing audio and MIDI
- Different types of AUs are designed for broad categories of functionality

AudioUnit—How It Works

- A component interface—so specifies an API that all AUs must support
- First step—find the different AUs on the system with `FindNextComponent`
- Open the unit you want with `OpenAComponent`
  - Essentially creates an instance of the class that is specified by the component type
- All of the AU API are directed to these instances:
  - `typedef ComponentInstance AudioUnit;`

AudioUnit—How It Works

- Next you configure its state
  - This can be simple or complex and often depends on the AU itself
- AudioUnit Get/Set Property
  - Generic mechanism for both finding out the current state of an AudioUnit and altering that state

AudioUnit—How It Works

- `AudioUnitInitialize`
  - Once the major state of an AudioUnit is established, the host will Initialize it
    - Allocates whatever resources the AU needs to do its work
    - Once initialized, the AU is now ready and able to do its work
- `AudioUnitRender`
  - This is called in order to have the AU produce its audio data

AudioUnit—How It Works

- Where does it get input data from?
  - An AU can either be provided data from
    - (1) an application supplied callback
    - (2) Another AU that it is connected to
- When `AudioUnitRender` is called the AU will get its input data, process it and provide you with its output

AudioUnit—How It Works

- What about real-time control of an AU's processing?
- AudioUnit Parameters
  - An AU publishes a list of parameters that can typically be altered in real-time to affect its processing constraints
    - For instance:
      - Delay time
      - Filter cut off frequency
      - Etc.

Apple FX AudioUnits

- Jaguar
  - Various filters
    - High/low/bandpass...
  - Reverb
  - Digital Delay
  - PeakLimiter
- Panther
  - Multi-band compressor

AudioUnit—How It Works

- So, an AU has the following features
  - Create one!
  - State management—AU Properties
  - Resource allocation—AU Initialization
  - Rendering—AU Render API
  - Rendering Control—AU Parameters
- What else do you need?

AudioUnit Views

- AudioUnit can publish Custom Views
  - "Generic" view will work with most AudioUnits
    - Basic "slider" type of view
- Jaguar—Carbon UI
- Panther—add support for Cocoa UI

AudioUnit Controller Support

- AUParameterListener
  - Jaguar
    - Notification service for changes in parameter values
- AUEvent Listener
  - AUProperty
    - Mechanism for notification of changes in state (for instance, data values)
      - “Public” or “Private” state can be shared between the AU and its View

AudioUnit Controller Support

- AUEvent Listener
  - Begin/End Parameter Ramp state
    - In Jaguar this was defined as a selector on the Carbon View Component
      - No longer requires specific view support
      - The start and end of a user's "Gesture" to alter the value of a parameter, whatever source this comes from
- Real-Time Thread safe notification mechanism

“Traditional” Audio Plugins

- 2 categories
  - Effects
    - Take Audio Signals as Input, apply some kind of effect to the audio (for e.g., Delay, Filter, etc.), then output the processed audio
  - Instruments
    - Take MIDI Data as a control source and generate audio in response to the MIDI commands
    - Software Synthesisers

AudioUnit MusicDevice

- <AudioUnit/MusicDevice.h>
- Extends the AudioUnit API
  - Adds semantics for:
    - Start/end notes
    - Control of playing notes
  - MIDI and Extended Protocols supported

AU Offline Unit

- New for Panther
- Similar to an effect unit but:
  - Doesn’t have real-time constraints
    - Typically deals with file data. Eg. “Reverse” AU
  - 2 Render phases
    - Analyze
    - Render
- Developed in partnership with Waves
  - http://www.waves.com/

“Non-Traditional” AU

- Mixer Units
  - Jaguar
    - Stereo Mixer
      - Mono or stereo inputs mixed to a single stereo output
    - 3DMixer
      - Single output of either 2 (stereo), 4 (quad) or 5 (5.0) channels
      - Mono or stereo inputs
        - Panned and localization within a 3D space
        - Rendering has a number of on/off options:
          - Reverb, Doppler, Distance Filtering, Panning Modes

Demo

3DMixer

AU—Mixers

- Panther
  - Matrix mixer
    - N->M channel mixer with X->Y in and out “busses”
  - All mixers have metering capability

AU Converter AudioUnit

- Presents the functionality of the AudioConverter within the AU world to provide flexible conversion of formats
  - Sample rate converter
  - Bit depth converter
    - Float-int
    - Channel mapping
- All this functionality is presented in one AU
  - Configured by properties
    - Typically the in and out formats

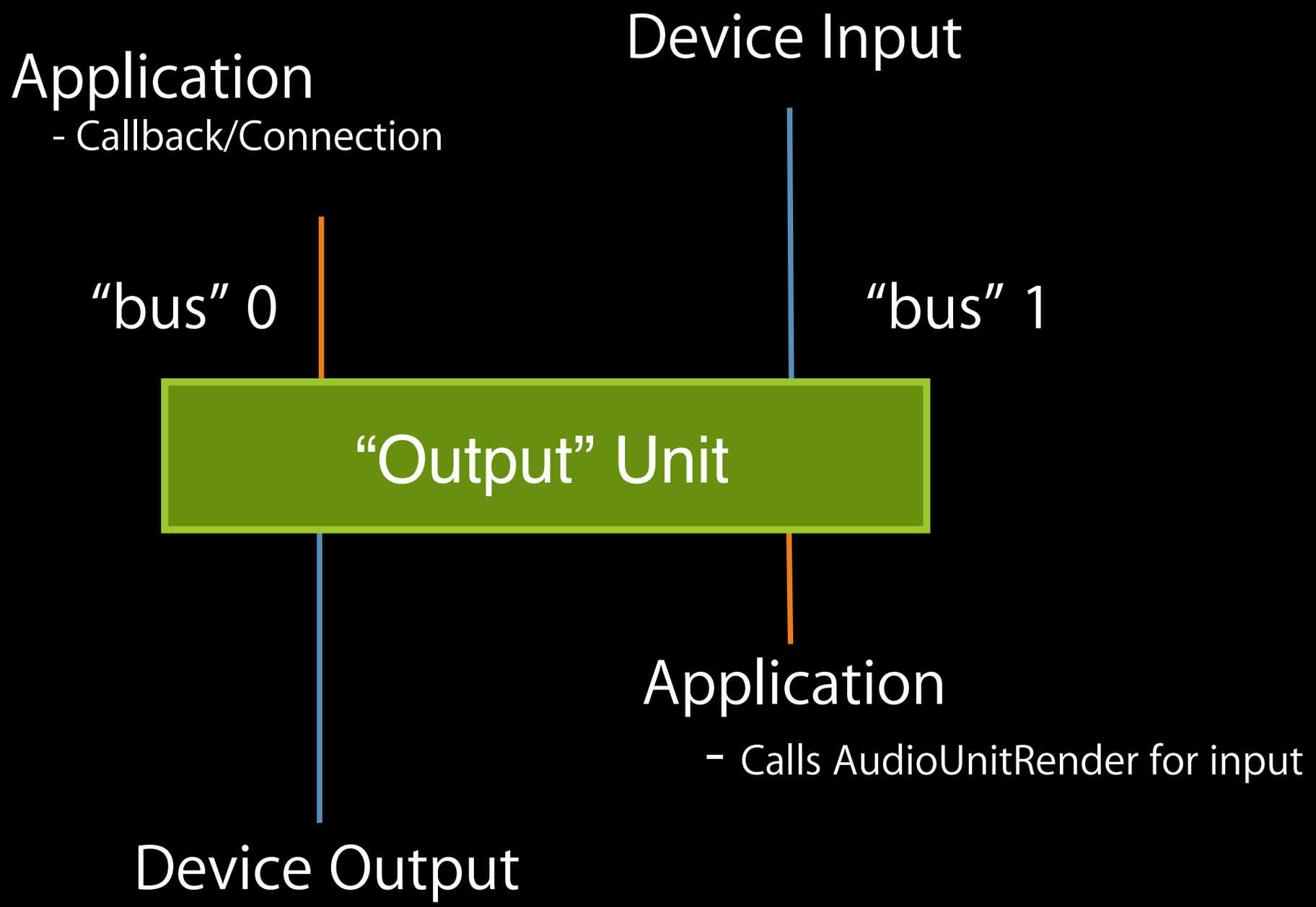
“Output” Units

- Greatly simplifies the interface to an Audio Device
  - No additional latency
  - Runtime cost?
    - Embeds the functionality of the AUConverter, so cost is determined by the difference between the format that the client renders and the format of the device
    - Can be managed by the user—no “hidden fees”

“Output” Units

- Jaguar
  - Only do output
- Panther
  - Now can do input as well...
  - Element Zero
    - Output bus (if present) on the device
  - Element One
    - Input bus (if present) on the device

“Output” Units I/O



AUGraph

- Manages the connections between a collection of AudioUnits
  - State management
    - Open/initialized/running
  - Can dynamically update its state (whilst running)
    - Make/break connections

MusicSequence

- A collection of MusicTracks
  - Tempo Track
    - Tempo and Time Signature Events
  - Event Tracks (1 to N)
    - MIDI Events
      - Note, Control, SysEx, Meta
    - User Events
    - AUParameter Events
      - Inc. support for Ramp Events

        Change from val=x to val=y over Time=t

MusicSequence

- Can create from/Save To MIDI files
- MusicTracks
  - Can iterate over the events in the track
  - Edit events and tracks
  - Loop, Mute/Solo, Start-Offset
- Target tracks events to:
  - An AudioUnit in the Sequence's AUGraph
  - A MIDI Endpoint to output to a MIDI Device

MusicPlayer

- So you want to play all this!
- Start/Stop
- Preroll
  - Prepares state so the Start/Stop is “instantaneous”
- Get/Set Time
- Panther
  - Scale the playback rate
    - Faster/slower playback of the events

Connecting It All Together

MusicPlayer

AudioFile

MusicSequence

MusicTracks

Tempo

Events

AudioConverter

AudioCodec

AUGraph

DLSSynth

AUFilter

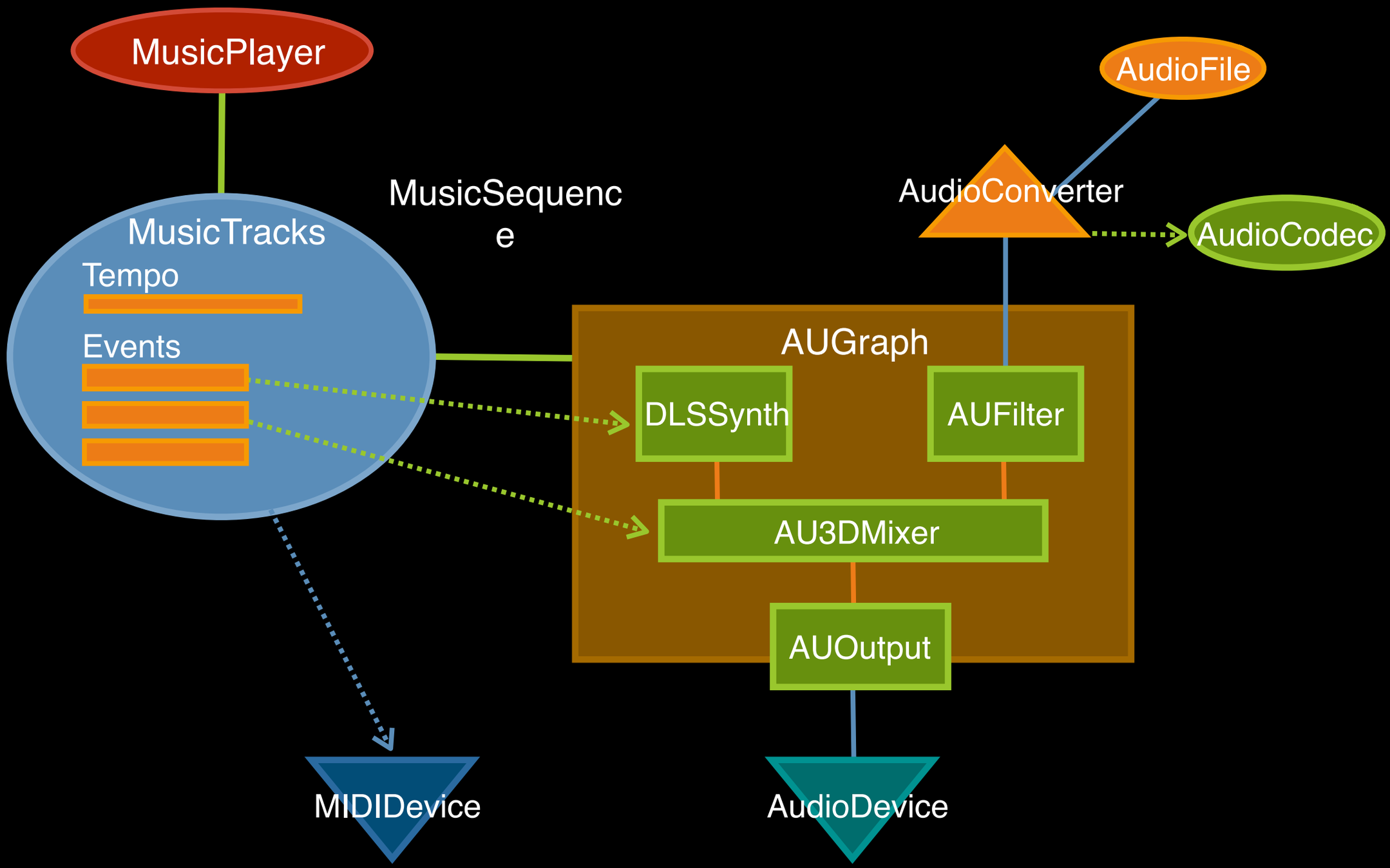
AU3DMixer

AUOutput

MIDIDevice

AudioDevice

Connecting It All Together



Connecting It All Together

- CoreAudio provides an architecture of components and services that can be assembled by applications
  - Audio/MIDI Device Access
  - Effects/Software Synthesis
  - Mixing Audio sources
  - Playing Sequences of events
  - Audio Conversion operations
  - Audio File operations

What All of This?

- Don’t panic!
- CoreAudio is set up in such a way that you only need to use those pieces that you need

Summary

- CoreAudio—HAL
  - Audio Drivers
- CoreMIDI
- AudioUnits
  - Extensibile Audio DSP
  - AudioCodecs
- AudioToolbox
  - Format Conversion
  - File Services
  - Processing graphs and event scheduling

Wrap Up

Roadmap

100 IOKit: Kernel Extension Programming Techniques Designing KEXTs for optimal performance	**Russian Hill** **Mon., 3:30pm**
405 Audio Formats	**Russian Hill** **Tues., 5:00pm**
711 Audio and QuickTime	**Russian Hill** **Wed., 9:00am**
FF003 Audio	**North Beach** **Wed., 10:30am**

Who to Contact

Craig Keithley
I/O and Security Technology Evangelist
keithley@apple.com

Craig Linssen
Music and Audio Developer Relations Manager
craig@apple.com

Nick Thompson
CPU Audio SW Manager—Device Driver Development
nthompson@apple.com

William Stewart
Core Audio Manager
bills@apple.com

Core Audio Mailing List
coreaudio-api@lists.apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

Core Audio Documentation
http://developer.apple.com/audio/

Q&A

Craig Linssen
Music and Audio Partnership Manager
craig@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference